Technology Mapping for Minimizing Gate and Routing Area

Aiguo Lu   Guenter Stenz   Frank M. Johannes

Institute of Electronic Design Automation
Technical University of Munich
80290 Munich, Germany

Abstract

This paper presents a technology mapping approach for the standard cell technology, which takes into account both gate area and routing area so as to minimize the total chip area after layout. The routing area is estimated using two parameters available at the mapping stage; one is the fanout count of a gate, and the other is the "overlap of fanin level intervals". To estimate the routing area in terms of accurate fanout counts, an algorithm is proposed which solves the problem of dynamic fanout changes in the mapping process. This also enables us to calculate the gate area more accurately. Experimental results show that this approach provides an average reduction of 15% in the final chip area after placement and routing.

1 Introduction

In the hierarchical design of digital systems, the interconnect information is only available at the low level of the design process. While the lower level design has more detailed interconnect information, it is limited in changing the netlist of a circuit. As VLSI fabrication technologies are entering Gigahertz frequencies and device dimensions are shrinking to the deep submicron region, the interconnect area and interconnect delay are more and more dominating the whole chip area and the circuit delay. Therefore, it is crucial to take into account the interconnect information at higher levels of the design process, such as technology mapping, where there exists more freedom to restructure the circuit [1].

Because of lacking the accurate geometrical information of gates and nets, the interconnect information can only be estimated at higher levels of the design process. For this reason, many previous technology mapping tools concentrate on minimizing the gate area as an approximation to area optimization. Recently it has been realized that a solution to the challenge posed by the new IC technologies is to improve the interaction between logic synthesis and layout tools. Addressing interconnect issues at the technology mapping stage has become a necessity because structural and functional flexibilities available in technology mapping can indeed be exploited to greatly impact the routing cost of the circuit. For area optimization as considered in this paper, it requires to shift the focus of technology mapping tools from traditional "gate area optimization" to "chip area optimization".

The main contribution of this paper is to consider both gate area and routing area so that the mapped circuits have smaller chip areas after layout. The routing area is estimated using the parameters extracted from the network structure; one is the fanout count and the other is the overlap of fanin level intervals. The fanout count is an important factor affecting the mapped circuits because it is related to not only the estimation of routing area but also the calculation of gate areas in the tree based mapping approaches. Traditional mapping algorithms take the fanout count in the original subject graph and use it as a constant. However, the fanout count is changing dynamically in the mapping process. Therefore, we propose an algorithm to predict and update the fanout changes so as to calculate the gate area and also to estimate the routing area more accurately.

The rest of the paper is organized as follows. Section 2 briefly reviews the previous achievements. Section 3 describes our approach. Section 4 presents the experimental results to evaluate the proposed approach. Finally, conclusions are made in Section 5.

2 Background

There have been several research achievements considering the routing cost during logic synthesis. Pedram and Bhat [2] proposed a layout driven synthesis approach which produces a logic synthesis solution and a "companion" placement solution simultaneously. The technique used here is to extract common divisors considering the normal cost function of literal savings and a cost function of estimated interconnect lengths. To compute the cost function of interconnect lengths, the sub-circuits related to each node function are placed before extraction. During extraction, the wire length needed to connect the extracted divisor (new node) and its fanout nodes in the network before and after extraction are calculated. More recently, Vaishnav and Pedram [3, 4] proposed approaches that consider the routing cost during logic extraction. Two simple routing cost functions were proposed in [3]; one was based on the minimization of fanout range and the other was based on the minimization of fanout range overlap. Another routing cost function that characterizes relative net lengths based on their pin-count was proposed in [4], in which the relative net length of a net with n pins is estimated to be $\frac{\sqrt{n}+1}{2}\frac{n-1}{n+1}$. These routing estimations together with the literal saving cost function were used for finding "good" divisors.

While all above approaches are in the area of multi-level logic optimization, Pedram and Bhat [5] also proposed a technology mapping tool called *Lily*, which estimates the interconnect dependent contributions to circuit area by referring to a dynamically updated global placement of the Boolean network. Like in [2], technology mapping was incorporated with the global placement so that the information of placement can be used to estimate the routing cost. The routing cost is calculated according to the wire length extracted from the placement information.

In this paper, we propose an approach that performs technology mapping for minimizing both gate area and routing area. Unlike *Lily* [5] which incorporates technology mapping with placement, we minimize the chip area using the following new techniques.

3 Mapping Techniques

Technology mapping can be formulated as a DAG (Directed Acyclic Graph) covering problem [6]. The Boolean network to be mapped is represented as a DAG, called the **subject graph**, using a set of chosen base functions (e.g., NAND gate and inverter). Similarly, gates in the library are represented as DAGs, called the **pattern graphs**, using the same base functions. Technology mapping is then the problem of finding an optimal cost covering of nodes in the subject graph using available patterns. There have been quite a lot of research achievements in this area [7, 6, 8, 2, 5, 9], from which it is found that the tree covering approach proposed by Keutzer [6] is an acceptable approximation to the general graph covering approaches for cost functions involving area, delay, and combinations thereof [8]. In the tree covering approach, the subject graph is partitioned into a forest of trees by breaking it up at each multiple-fanout point. Its main problem is that it does not allow any match across multiple-fanout points, and thus no tree overlap is allowed. In [10, 8], an alternative approach was used, in which the library is allowed to have non-tree patterns and the subject graph is a general DAG. Starting from the primary inputs, subject DAG nodes are traversed in a depth-first manner. All patterns that match at a node are enumerated, and the minimum cost match is stored for the node. This approach works well for most circuits [8], and so it is used in this paper.

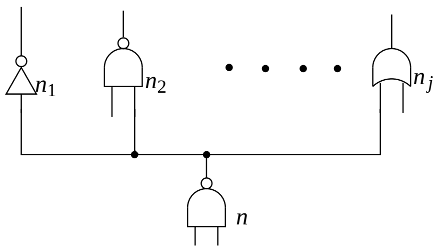

Figure 1: Effect of multiple-fanout points.

Definition 1: A **match** at node n is a part of the subject graph rooted at node n and covered by one of the pattern graphs.

Most of the previous mapping algorithms for area optimization are focused on minimizing the gate area. So in the cost function which directs the selection of best matches, only gate area is involved. Improvements are to be expected if both gate area and routing area are considered in technology mapping. Therefore, we use two cost functions to represent the cost of a match m, one is the gate area cost and the other is the estimated routing area cost.

Let $SGA(m)$ be the $\underline{s}$ummation of $\underline{g}$ate $\underline{a}$rea (or gate cost) at match m, and $SRA(m)$ be the $\underline{s}$ummation of estimated $\underline{r}$outing $\underline{a}$rea (or routing cost) at match m. For the strictly tree-based mapping approach, they can be calculated as follows:

$$SGA(m) = gate(m) + \sum_{m_i \in F_{in}(m)} SGA(m_i) \quad (1)$$

$$SRA(m) = routing(m) + \sum_{m_i \in F_{in}(m)} SRA(m_i) \quad (2)$$

Here, $gate(m)$ refers to the area of the physical gate corresponding to match m, $routing(m)$ represents the estimated routing area of match m, and $F_{in}(m)$ represents the set of inputs to match m. The calculation of gate cost $SGA(m)$ is simple, and similar to that in [7]. The estimation of routing cost $SRA(m)$ will be discussed later.

In our approach, non-tree patterns are allowed. So the propagation of costs from the multiple-fanout points should be considered carefully. As shown in Figure 1, suppose the gate cost of the best match m_n at node n is $SGA(m_n)$. $SGA(m_n)$ will be propagated to each of n's fanouts when they are being mapped. Let $SGA(m_{n_i})$ be the gate cost at n_i and $gate(m_{n_i})$ be the area of the gate of match m_{n_i}, $i \in \{1, 2, \cdots, j\}$. The heuristic to calculate $SGA(m_{n_i})$ is

$$SGA(m_{n_i}) = gate(m_{n_i}) + 1/j \times SGA(m_n) + S_i \quad (3)$$

where S_i is the summation of the gate areas propagated from other inputs of gate n_i. This heuristic is known to give good results in most cases [8]. So we will use this heuristic to calculate $SGA(m_{n_i})$ and also $SRA(m_{n_i})$. Apparently, the fanout count j will affect the cost of the match, and thus the choice of the best match at n_i. In the previous mapping algorithms for area optimization, the fanout count in the subject graph is taken and used as a constant when calculating the propagation of gate areas. However, the fanout count is most likely changing in the mapping process.

In the following, the estimation of the routing cost will be discussed first. Then the dynamic change of fanout counts is investigated, and the techniques to deal with this are proposed.

3.1 Estimation of routing cost

Many parameters contribute to the routing cost. In [11], 20 parameters were used to find their influence to the net length based on a large set of benchmark examples. It is shown that the fanout count is the most important factor affecting the net length. This is reasonable because a multi-pin net usually spans a longer distance than a 2-pin net and it also complicates the subsequent layout process. Therefore, the fanout count is used as a factor in our routing cost function.

It is shown in [3] that minimizing the fanout ranges of extracted divisors (nodes) in multi-level logic optimization will help to distribute nets uniformly on the chip, which reduces the routing congestion, and so the routing cost. In technology mapping, the fanout range of a node is unknown when this node is being mapped. However, the uniform distribution of nets can also be achieved if it is possible to bring all fanin nodes of a node close to each other according to the geometrical or topological distance. While the geometrical distance is unknown at this stage, a topological distance is used here, which is based on a concept called the overlap of fanin level intervals. The level of a node is known as the number of nodes along the longest path from primary inputs to this node.

Definition 2: Assume m is one of the matches found at the node being mapped, and $F_{in}(m)$ is the set of fanin nodes to match m. For each node $n \in F_{in}(m)$, the level of n, denoted as $LEV(n)$, is known in the mapped network. Let MAX_LEV be the maximum level of nodes in $F_{in}(m)$. Then the overlap of fanin level intervals for m, denoted as $OVL(m)$, is defined as $OVL(m) = \sum_{n \in F_{in}(m)} (MAX\_LEV - LEV(n))$.

Figure 2 shows a simple example how to calculate $OVL(m)$. Let the inputs to a match m be a, b, c, and d. Fanin nodes a and b are primary inputs whose level is 0, and fanin nodes c and d are intermediate nodes whose levels are 7 and 4, respectively. Then the overlap of fanin level

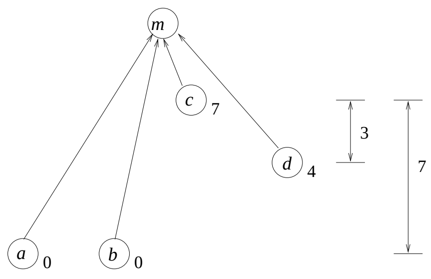

Figure 2: Illustration of overlap of fanin nodes.

intervals for match m is $7+7+3 = 17$.

Definition 3: Assume m is one of the matches at node n, the routing area caused by this match m is estimated to be $routing(m) = F_c(n) \times F_{cost} + OVL(m) \times OVL_{cost}$, where $F_c(n)$ is the fanout count of node n, $OVL(m)$ is the overlap of fanin level intervals for match m at n, F_{cost} is the weight of fanout cost, and OVL_{cost} the weight of overlap cost. F_{cost} and OVL_{cost} are related to the implementation technology and the circuit size, which will be given in Section 4.

With the above definition, we are able to determine the routing cost of a match m, $SRA(m)$. In the mapping process, both the gate cost $SGA(m)$ and the routing cost $SRA(m)$ are calculated for each match m according to equations (1)–(3). Assume bm_n is the current best match of node n found so far, and its costs are $SGA(bm_n)$ and $SRA(bm_n)$. If, for a new match m, either of the two conditions

1. $SGA(m) \leq SGA(bm_n)/\alpha$ and $SRA(m) < \beta SRA(bm_n)$
2. $SGA(m) < \alpha SGA(bm_n)$

is satisfied, match m is considered to be better than bm_n, and will replace bm_n to become the new best match for node n. Here, α $(0 < \alpha \leq 1)$ and β $(0 < \beta \leq 1)$ are two parameters used to tradeoff the gate cost and the routing cost, and they will be discussed and given in the experimental section.

3.2 Fanout changes

The fanout count is used as a key factor in our routing cost estimation. However, the fanout count is changing in the mapping process. Suppose that a match at node n of Figure 3 is a gate **blf00101** with inputs a, b, and c. While the original fanout count of node b in the subject graph is 3, it becomes 2 if this gate is selected as the best match at node n.

The fanout count which really affects the chip area is the one in the mapped circuit rather than that in the subject graph. An ideal solution is to use the fanout count in the mapped circuit to calculate the propagation of gate areas and routing areas as well as to estimate the routing area. The problem is that this count is not available until the mapping process terminates. In the following, we

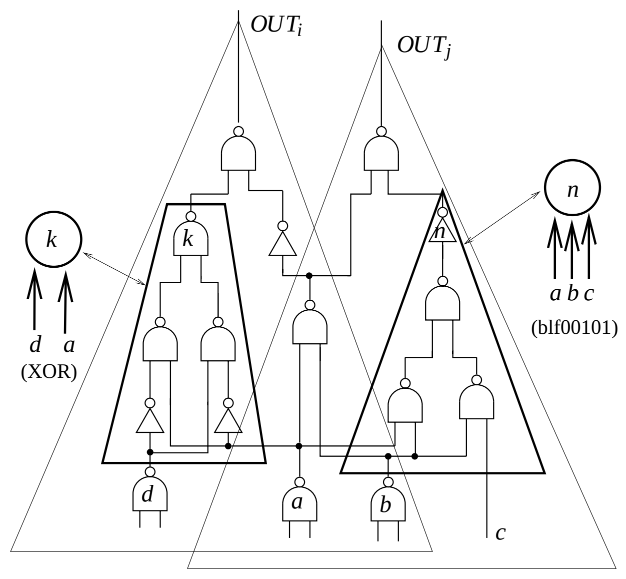

Figure 3: Dynamic changes of fanout count.

propose an algorithm to solve this problem.

3.2.1 Logic cone based mapping routine

To deal with the fanout changes, we first apply a mapping strategy that the mapping proceeds one logic cone followed by another. To do so, we partition the subject graph into a set of logic cones, each of which corresponds to one primary output. As shown in Figure 3, there are two logic cones; one corresponds to OUT_i and the other to OUT_j. We map one logic cone first, traversing from its transitive primary inputs to the primary output. All the changes of the fanout count are calculated and updated. Then we select another logic cone and map it in a similar way. The updated fanout count of each node is used in calculating the gate cost and estimating the routing cost, and it may be further updated when mapping the current logic cone.

As an example, assume the logic cone associated with the primary output OUT_j in Figure 3 is being mapped now. Let the best match at node n be gate blf00101. While the fanout count of node b in the subject graph is 3, its actual value is 2 if this gate is selected. So this fanout count is updated and will be used when mapping other logic cones.

By mapping one logic cone at a time, the more accurate fanout count can be used. However, the fanout count updated in this way is still not exactly the same as the one in the mapped graph. This can be illustrated in Figure 3 as well. Assume the logic cone associated with OUT_j has been mapped, and the logic cone related to OUT_i is being mapped now. Let the best match at node k be an XOR gate with inputs d and a. In this case, the fanout count of node

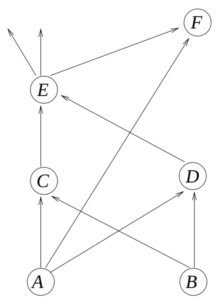

Figure 4: An example of the reconvergent paths.

a is reduced from 4 to 3. Recall that the fanout count of a was considered to be 4 when the logic cone related to OUT_j was being mapped. This implies that the area cost calculated in the previous logic cone is not correct anymore. So we propose another technique to further consider the fanout changes.

3.2.2 Prediction of fanout changes

Although the exact fanout count can be obtained only after the whole mapping process terminates, there are some features we can use to predict fanout changes. We restrict our concerns to the multiple-fanout nodes because only these nodes will affect the propagation of gate area and routing area. It is easy to note from Figure 3 that the fanout count of a multiple-fanout node may decrease if there are reconvergent paths from the multiple-fanout node to the node being mapped.

Definition 4: A pair of reconvergent paths associated with node n is defined as two paths which start from n and converge to another node.

Figure 4 shows an example of reconvergent paths. It can be seen that paths $A-D-E$ and $A-C-E$ are a pair of reconvergent paths. Similarly, paths $B-D-E$ and $B-C-E$ are a pair of reconvergent paths, and paths $A-C-E-F$ and $A-F$ are another pair of reconvergent paths. By determining the number of reconvergent paths, we predict the potential changes of fanouts during the mapping process. Recall that our mapping approach is not a real DAG-based covering approach, but allows non-tree patterns in the pattern graphs. Therefore, to reduce the search space, we only consider those reconvergent paths within a tree. Under this assumption, the reconvergent paths such as $A-C-E-F$ and $A-F$ are not considered to contribute to the fanout decrease of A because the path $A-C-E-F$ passes through another multiple-fanout node E. Another condition is that the maximum number of pairs of reconvergent paths at a multiple-fanout node should be less than its fanout count. This is because there is no match across the multiple-fanout node in our mapping process. So in our approach, a simple search step is applied to each multiple-fanout node of the subject graph. Starting from one multiple-fanout node, the number of pairs of reconvergent paths within a tree is determined. If this number is N, we predict that the fanout count of node n is $F_c(n) - N$, where $F_c(n)$ represents the original fanout count of node n in the subject graph. For the subject graph shown in Figure 4, it is easy to determine that N is equal to 1 for nodes A and B. So the predicted fanout count of nodes A and B is 2 and 1, respectively. This predicted fanout count will be used to estimate the routing cost and to calculate the propagation of gate areas and routing areas. If the predicted fanout count differs from the actual value during the logic cone based mapping process, the fanout count of related nodes will be modified.

4 Experimental Results

The proposed approach has been implemented under the SIS [10] environment. The inputs to the mapper are the circuits optimized using the SIS optimization script *script.rugged*. These optimized circuits are then fed into the SIS mapping algorithm and the proposed algorithm. The library used for the experiments is *stdcell2_2.genlib*, for which we have the physical descriptions for placement and routing. The proposed mapping approach is called MIGO (*Mapping for both Interconnect and Gate Optimization*). Both SIS and MIGO used the same decomposition algorithms and base functions (NAND gate and inverter) to construct subject and pattern graphs. Two sets of mapped results were fed into the same layout tool to get the final chip area for comparison. GORDIAN [12] was used for placement, TimberWolf [13] for routing.

For the SIS mapping, we use the recommended command for area optimization (*map -m 0.0*). For the MIGO mapping, the tradeoff parameters should be determined during the experiments. In our experiments, the default values of α and β were set to be 0.95 and 0.7 respectively, which means that about 5% gate cost increase is allowed when selecting a current match to be the best match as long as the routing cost of this match is over 30% less than the original one. Both α and β are dependent on the implementation technologies and the sizes of circuits. In our experiments, the implementation technology was not changed. The influence of different circuit sizes was adapted to the estimation of routing cost. So we used the default values in the whole set of experiments. For the parameters related to the routing cost, we determine them based on the experiments, and set $\{F_{cost}, OVL_{cost}\} = \{7, 0.5\}$ for the big circuits (*des*, *C5315*, and *C6288*) and set $\{F_{cost}, OVL_{cost}\} = \{2, 0.3\}$ for other circuits.

The experimental results are shown in Table 1. The columns labeled with *g_area*, *delay*, and *c_area* represent the total gate areas, the circuit delay, and the final chip area. The total gate areas were obtained after technology mapping, the delay was calculated using the Elmore delay formula based on the star model after placement, and the chip area was provided by the router. As shown in column 10 of Table 1, the circuits produced by MIGO require 15% smaller chip area, on average, when compared with those produced by SIS. Moreover, the delay of MIGO results was not deteriorated when achieving area reduction. While 3% delay reduction (column 9) was mainly caused by the example *des*, it should be noted that, on average, the delay of circuits mapped by MIGO is still less than that of circuits mapped by SIS even without considering *des*.

A careful investigation of Table 1 shows that less gate area does not always lead to less chip area (e.g., *C5315* and *Z9sym*). This justifies our claim that technology mapping considering only the gate area is not sufficient for achieving the minimal chip area. Column 8 also shows that MIGO provided mapped circuits with less gate area (6% less). This was mainly achieved by the fanout prediction. The fanout prediction enables us to calculate the propagation of gate areas and routing areas more accurately, thus directing the correct selection of the best match. Comparing column 8 with column 10 shows that the chip areas were reduced more than the gate areas. This implies that the routing cost is an important factor affecting the chip area and that MIGO provides mapped circuits which require less routing cost.

It is difficult to present a fair comparison between MIGO and other contributions that also consider the routing cost in logic synthesis because most of them are in the area of multi-level logic optimization [3, 4, 2] except the tool *Lily* [5] which combines technology mapping and placement. However, we cannot compare MIGO with *Lily* directly because it was not reported what kind of optimized circuits Lily used. Nevertheless, if only the results after layout are concerned, we feel confident to say that MIGO provides better results than *Lily* (*Lily* requires average 5% smaller chip area when compared with MIS [14], the former version of SIS, on the basis of 15 examples using the same placement and routing tools as we used.) and also other approaches in [3, 4, 2]. This is mainly because MIGO considers the fanout changes in the mapping process so that the estimation of routing cost and also the calculation of gate cost become more accurate.

5 Conclusions and Future Work

An approach for minimizing chip area in technology mapping was presented in this paper. Two methods were proposed; one is to consider the routing cost and the other is to predict the fanout changes. By considering the routing cost, the mapped results can become a better starting point for layout. The routing cost function is estimated according to the fanout count and the overlap of fanin level intervals. By predicting the fanout changes, the calculation of gate

Name	SIS			MIGO			Percentage		
	g_area	delay	c_area	g_area	delay	c_area	g_area	delay	c_area
z4ml	656	8.80	744×199	456	7.85	536×159	0.695	0.892	0.576
f51m	1448	22.23	809×427	1208	21.1	689×347	0.834	0.949	0.692
rd73	1048	10.34	609×323	1032	11.28	609×283	0.985	1.091	0.876
rd84	2216	13.94	841×623	2232	13.33	849×519	1.007	0.956	0.841
5xp1	1872	21.63	721×559	1776	18.35	681×487	0.949	0.848	0.823
cm150a	720	6.15	792×215	672	6.07	744×207	0.933	0.987	0.904
Z9sym	3424	14.93	1009×891	3440	13.94	985×803	1.005	0.934	0.880
b9	2072	7.67	817×575	1840	7.44	737×527	0.888	0.970	0.827
apex2	3992	11.22	1153×867	3752	11.86	1049×739	0.940	1.057	0.775
ex5	5032	24.16	1345×1119	4808	22.71	1153×1127	0.955	0.940	0.863
too_large	5768	13.99	1707×617	5672	14.16	1662×586	0.983	1.012	0.925
duke2	7480	24.33	1625×1507	7336	25.44	1561×1491	0.981	1.046	0.950
C432	3064	36.05	961×915	3056	38.03	953×787	0.997	1.055	0.853
C880	6256	30.06	1433×1151	6224	33.55	1473×1023	0.995	1.116	0.914
C1355	6960	18.54	1393×1099	6504	17.42	1289×987	0.934	0.940	0.831
C1908	7632	28.67	1537×1187	7032	26.71	1393×1115	0.921	0.932	0.851
i9	10248	41.53	2001×2143	8944	39.33	1849×2019	0.873	0.947	0.871
dalu	14728	40.12	2713×1295	13976	39.71	2536×1212	0.949	0.99	0.875
i8	16240	31.84	2746×1774	15752	30.88	2595×1608	0.970	0.970	0.857
C5315	24616	26.17	2825×2639	25000	27.55	2737×2479	1.016	1.053	0.910
pair	25808	37.60	3524×2197	25824	36.55	3515×2119	1.001	0.972	0.962
C6288	49728	88.66	4647×2298	43088	96.06	4233×2280	0.866	1.083	0.904
des	51656	60.64	4697×6179	50528	32.176	4609×5403	0.978	0.531	0.858
average							0.94	0.97	0.85

Table 1: Comparison of experimental results.

areas and the estimation of routing cost become more accurate, thus directing the mapping to achieve better results.

We are now enhancing MIGO to involve delay optimization. As the circuit delay is path-oriented, new routing cost functions and mapping strategies are being studied.

Acknowledgments

The authors would like to thank Prof. Kurt Antreich at the Technical University of Munich for his continuous support to their work. Thanks also to Christian Legl, Klaus Eckl and Bernhard Rohfleisch for valuable discussions.

References

[1] M. Pedram, R. Bushroe, R. Camposano, G. D. Micheli, A. Domic, C.-P. Hsu, and M. Jackson, "Panel: Physical design and synthesis : Merge or die," in *ACM/IEEE Design Automation Conference (DAC)*, June 1997.

[2] M. Pedram and N. Bhat, "Layout driven logic restructuring/decomposition," in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 1991.

[3] H. Vaishnav and M. Pedram, "Minimizing the routing cost during logic extraction," in *ACM/IEEE Design Automation Conference (DAC)*, pp. 70–75, 1995.

[4] H. Vaishnav and M. Pedram, "Logic extraction based on normalized netlengths," in *IEEE International Conference on Computer Design (ICCD)*, pp. 658–663, Oct. 1995.

[5] M. Pedram and N. Bhat, "Layout driven technology mapping," in *ACM/IEEE Design Automation Conference (DAC)*, pp. 99–105, June 1991.

[6] K. Keutzer, "DAGON: Technology binding and logic optimization by DAG matching," in *ACM/IEEE Design Automation Conference (DAC)*, pp. 341–347, June 1987.

[7] R. Rudell, *Logic synthesis for VLSI design*. PhD thesis, University of California, Berkeley, 1989. Ph.D. Thesis.

[8] V. Tiwari, P. Ashar, and S. Malik, "Technology mapping for low power in logic synthesis," *INTEGRATION - the VLSI journal*, vol. 20, pp. 243–268, July 1996.

[9] R. Rudell, "Tutorial: Design of a logic synthesis system," in *ACM/IEEE Design Automation Conference (DAC)*, pp. 191–196, June 1996.

[10] E. M. Sentovich, K. J. Singh, C. Moon, H. Savoj, R. K. Brayton, and A. Sangiovanni-Vincentelli, "Sequential circuit design using synthesis and optimization," in *IEEE International Conference on Computer Design (ICCD)*, pp. 328–333, Oct. 1992.

[11] H. Jyu and S. Malik, "Prediction of interconnect delay in logic synthesis," in *European Design and Test Conference (ED&TC)*, pp. 411–415, Feb. 1995.

[12] G. Sigl, K. Doll, and F. M. Johannes, "Analytical placement: A linear or a quadratic objective function?," in *ACM/IEEE Design Automation Conference (DAC)*, (San Francisco), pp. 427–432, 1991.

[13] K. Lee and C. Sechen, "A new global router for row-based layout," in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, (Santa Clara), pp. 180–183, 1988.

[14] R. K. Brayton, R. Rudell, A. Sangiovanni-Vincentelli, and A. R. Wang, "MIS: A multiple–level logic optimization system," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems CAD*, vol. 6, pp. 1062–1081, Nov. 1987.